Polytechnique Montréal

Département de génie informatique et génie logiciel

Cours INF1995:
Projet initial en génie informatique et travail en équipe

Travail pratique 8

Makefile et production de librairie statique

Par l'équipe

N° 2526

Noms:

Gergi Younis
Anthony Dentinger
Vincent Dandenault
Emir Khaled Belhaddad

Date:
1er Novembre 2016

Partie 1 : Description de la librairie

Tout d'abord, il nous faut préciser qu'un des deux groupes de deux de notre équipe **avait déjà rédigé un ensemble de classes et fonctions**, sans savoir qu'il s'agirait d'un travail demandé par la suite. Nous comprenons qu'un des objectifs de ce travail était de discuter en groupe des fonctionnalités à regrouper, mais il faut avouer que créer une librairie à partir de zéro alors qu'il en existe déjà une –et en grande partie testée, qui plus est– présente peu d'intérêt.

Notre librairie regroupe **environ toutes les fonctionnalités** des TP précédents, comme le montre le diagramme de classes suivant qui illustre les liens entre les classes de notre librairie, ainsi que les méthodes les plus importantes :

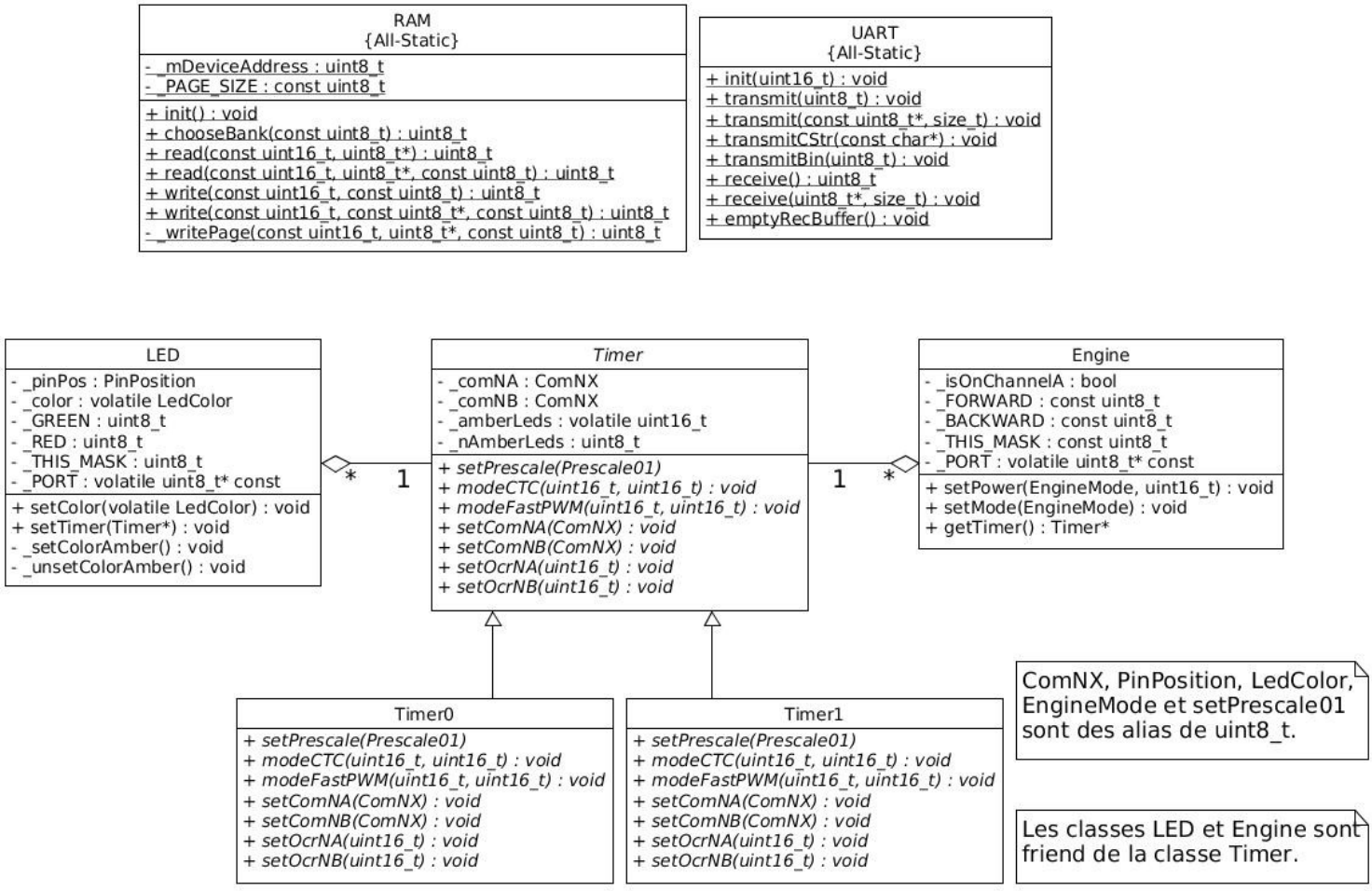

Fig. 1 – Diagramme de classes et méthodes principales de la librairie construite. Constructeurs, destructeurs et certaines autres méthodes non représentés.

La librairie a été développée avec une optique d'abstraction, donc de manière à pouvoir se concentrer sur la tâche à accomplir plutôt que sur l'utilisation de la librairie. Toutefois, la mémoire EEPROM du ATMega324PA et la fréquence d'oscillation du cristal d'horloge n'étant pas infinis, quelques détails méritent d'être mentionnés lors de l'utilisation de la librairie :

- La couleur ambre d'une LED contrôlée par un compteur N est **générée par l'interruption TIMER[N]_OVF**. L'utilisation de l'interruption OVF permet au compteur N de pouvoir gérer d'autres types d'actuateurs (comme des moteurs)

tout en générant la couleur ambre de cette LED. Pour que la couleur ambre puisse être générée, le programme dépendant de la librairie doit, d'une part, **permettre l'interruption OVF** du compteur impliqué (méthode `Timer::allowOVFI`) et, d'autre part, **définir la routine ISR** afin d'utiliser la fonction `switchAmberLedsColor`, déclarée dans le fichier `defaultISR.h`. Il est préférable de choisir un **compteur 16 bits** pour gérer les LED afin de réduire le temps de traitement des interruptions.

- Le fichier `main.cpp` **doit absolument contenir** des variables globales nommées exactement `timer0` et `timer1`, afin que l'éditeur de liens puisse lier ces variables avec celles utilisées dans `Engine.h` et `Engine.cpp`.
- La méthode `Engine::setMode` déconnecte le compteur du moteur et met une valeur fixe sur les broches, alors que `Engine::setPower` connecte le moteur au compteur et modifie la valeur de OCR[N][X] pour générer un signal PWM.

Notons que nous prévoyons de rendre les classes dérivées de `Timer` entièrement statiques –comme les classes `RAM` et `UART`–, puisqu'il ne fait pas de sens d'avoir plusieurs instances de ces classes. Des plus, les méthodes sur les compteurs pourront alors être utilisées n'importe où dans les programmes *fan-in*, sans avoir besoin de déclarer `extern Timer0 timer0; extern Timer1 timer1;` dans chaque fichier.

Partie 2 : Décrire les modifications apportées au Makefile de départ

Notre Makefile se base sur celui qui nous est fourni et comporte plusieurs modifications, amplement commentées, afin de l'adapter aux exigences de ce TP. Quatre modifications principales méritent toutefois d'être soulignées :

- Adaptation de la cible « all » pour accepter des librairies externes statiques, i.e. la librairie de ce TP. La présence d'une librairie externe permet de réduire le temps de compilation du code source du projet.
- Ajout de différentes variables qui vont prendre en charge les librairies et leurs chemins d'accès, i.e. LIBS et LIBS_PATHS, et l'archiveur utilisé ainsi que ses options, i.e. avr-ar et -crs.
- Modification des variables existantes utilisant des commandes spéciales de Make telles que « strip », qui retire les espaces en trop, et « patsubst », qui substitue un pattern par un autre fourni.
- Mise en place des cibles .PHONY « lib », pour produire une librairie .a, et « auto », pour faciliter les tests et déboggage pour de petites modifications de codes.

À noter que nous avons aussi ajouter des variables pour spécifier l'emplacement des fichiers sources à compiler ainsi que le chemin du dossier à créer (« gen ») pour y placer les fichiers générés lors de la compilation.

Notons enfin que nous avons séparé notre fichier Makefile original en un Makefile par répertoire et un Makefile_common dans lequel nous avons regroupé toutes les instructions communes à tous les types de compilation (archive ou exécutable).

Le rapport total ne doit par dépasser 7 pages incluant la page couverture.

Barème: vous serez jugé sur:

- *La qualités et le choix de vos portions de code choisies (5 points sur 20)*
- *La qualités de vos modifications aux Makefiles (5 points sur 20)*
- *Le rapport (7 points sur 20)*
  - *Explications cohérentes par rapport au code retenu pour former la librairie (2 points)*
  - *Explications cohérentes par rapport aux Makefiles modifiés (2 points)*
  - *Explications claires avec un bon niveau de détails (2 points)*
  - *Bon français (1 point)*
- *Bonne soumission de l'ensemble du code (compilation sans erreurs, …) et du rapport selon le format demandé (3 points sur 20)*